Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

Matej Brumen

ALGORITEM PREDVIDEVANJA POPLAVNIH OBMOČIJ Z UPORABO PODATKOV LiDAR

Diplomsko delo

Maribor, september 2013

Diplomsko delo visokošolskega študijskega programa

ALGORITEM PREDVIDEVANJA POPLAVNIH OBMOČIJ Z UPORABO PODATKOV LiDAR

Študent:	Matej Brumen
Študijski program:	Računalništvo in informacijske tehnologije (VS)
Smer:	/
Mentor:	red. prof. dr. Borut Žalik
Somentor:	doc. dr. Domen Mongus

Maribor, september 2013

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

Številka: E1041400
Datum in kraj: 25. 03. 2013, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 01/2010)
izdajam

SKLEP O DIPLOMSKEM DELU

1. **Mateju Brumnu,** študentu visokošolskega strokovnega študijskega programa RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE, se dovoljuje izdelati diplomsko delo pri predmetu Obdelava geometrijskih podatkov.

2. **MENTOR: red. prof. dr. Borut Žalik**
   SOMENTOR: doc. dr. Domen Mongus

3. **Naslov diplomskega dela:**
   ALGORITEM PREDVIDEVANJA POPLAVNIH OBMOČIJ Z UPORABO PODATKOV LiDAR

4. **Naslov diplomskega dela v angleškem jeziku:**
   AN ALGORITHM FOR FLOADING AREA PREDICTION USING LiDAR DATA

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih (dva trdo vezana izvoda in en v spiralo vezan izvod) ter en izvod elektronske verzije do 30. 09. 2013 v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:
red. prof. dr. Borut Žalik

UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO RAČUNALNIŠTVO IN INFORMATIKO

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv.

ZAHVALA

Zahvaljujem se mentorju red. prof. dr. Borutu Žaliku za pomoč in vodenje pri opravljanju diplomskega dela. Prav tako se zahvaljujem somentorju doc. dr. Domnu Mongusu. Zahvaljujem se tudi Niku Lukaču za nasvete pri opravljanju diplomskega dela. Posebna zahvala gre družini za podporo med tekom študija.

Algoritem predvidevanja poplavnih območij z uporabo podatkov LiDAR

Ključne besede: algoritmi, računalniška geometrija, enačbe plitvih voda, podatki LiDAR, 2.5D površje, realno-časovna simulacija tekočin, vizualizacija terena, vizualizacija tekočin.

UDK: 004.94(043.2)

Povzetek

Z napredovanjem računalniške tehnologije se zadnje čase pojavlja čim več različnih fizikalnih simulacij, med katere spadajo tudi simulacije dinamike tekočin. Enačbe Navier-Strokes opisujejo takšno gibanje tekočin v 3D prostoru, katerih diskretizacija in reševanje na CPU porabi veliko časa. Zato bi si želeli lažjo in hitrejšo metodo za tovrstne simulacije. V diplomskem delu smo predstavili implementacijo reševalnika enačb plitve vode, ki opisujejo gibanje tekočin v 2D prostoru. Reševalnik smo uporabili za predvidevanje poplavnih območij nad 2.5D mrežo zgrajeno iz 3D oblaka točk. Ta množica točk običajno predstavlja realna površja, ki so bila posneta s tehnologijo LiDAR. Tako je v razvitem orodju mogoče predvidet ogrožene predele, ki bodo poplavljeni v realnem času.

An algorithm for flooding area prediction using LiDAR data

Key words: algorithms, computer geometry, shallow water equations, LiDAR data, 2.5D grid surface, real-time fluid simulation, terrain visualization, fluid vizualization.

UDK: 004.94(043.2)

Abstract

With the recent advances of computer technology there are many different physical simulations, including simulations of the fluid dynamics. Navier-Strokes equations describe such fluid motion in 3D space. Solving Navier-Strokes equations on CPU is a time consuming task. Because of that simplified method is necessary for the CPU simulation. In this diploma thesis an implementation of the solver for shallow water equations (SWE) is presented. The solver for predicting flooded areas on 2.5D grid constructed from the point cloud is used. These points usually represent real surfaces that have been scanned with LiDAR technology. With our implemented application, the areas that will be flooded from a nearby water sources can be predicted in a real-time.

VSEBINA

1 UVOD

Poplave nas spremljajo že od samega začetka obstoja našega planeta. Velikokrat nastanejo kot posledica padavin skozi čas, ki so glavni razlog za naraščanje vode. Takšne poplave imenujemo hudourne poplave, ki v veliki meri za seboj pustijo razdejanje v urbanih in neurbanih področjih [11]. Območje velja za poplavljeno, kadar je njeno površje prekrito z določeno količino vode zaradi naraščanja bližnjih virov vode. Pred prihodom računalniše tehnologije je bilo praktično nemogoče napovedati poplavna območja. Vremenska napoved je takrat temeljila na spremembah zračnega tlaka. Z napredovanjem računalniške tehnologije so se razvile tudi tehnike napovedovanja vremenskih sprememb [2]. Odkrivanje predelov, ki bodo poplavljeni ob določeni količini vode, se da ugotoviti z različnimi simulacijami. Z analizo pridobljenih statističnih podatkov o preteklih poplavah ogrožene lokacije, lahko napovemo verjetnost in obdobje, v katerem bo prišlo do poplav. Tovrstna metoda je uporabna nad predeli, ki jim skoraj vsako leto grozijo poplave. Kollinger in soavtorji [19] so predstavili različico dane metode in jo implementirali v geografski informacijski sistem (GIS), natančneje v programsko orodje ArcGIS. Nad izbranimi podatki predhodnih poplav dveh večjih rek so izračunali vodne tokove. Nad podatki digitalnega modela terena (ang. Digital Terrain Model, DTM) so približno ocenili zgornje meje poplavljenih predelov. Al-Sabhan in soavtorji [12] so predstavili realno-časovno metodo predvidevanja poplavnih območij z uporabo GIS in svetovnega spleta, ki kot vhod prejme realno-časovne podatke o nevihti izbrane merilne postaje.

S simulacijo dinamike tekočin, ki jo podrobneje predstavimo v 2. poglavju, lahko predvidimo poplavljene predele izbranega terena v določenem časovnem trenutku na različnih lokacijah ob daljšem naraščanju količine vode. V diplomskem delu najprej predstavimo enačbe dinamike tekočin, ki so osnova za računalniške simulacije. V nadaljevanju predstavimo še metode in aplikacije tovrstnih simulacij. V 3. poglavju opišemo postopek implementacije našega orodja, kjer predstavimo ogrodje, grafični pogon, vhodne podatke, algoritem in upodabljanje rezultatov. V predzadnjem poglavju predstavimo rezultate in eksperimente našega diplomskega dela. V sklepu še povzamemo celotno delo.

1.1 Uporaba tehnologije LiDAR

V diplomskem delu smo se osredotočili na izvajanje simulacije z uporabo podatkov LiDAR (ang. Light Detection And Ranging). Tehnologija LiDAR uporablja laserski oddajnik, ki oddaja laserske impulze, da določi položaj objektov na površju [22]. Rezultat zračnega prebiranja LiDAR je množica ne strukturiranih 3D točk imenovanih oblak točk, shranjenih v

formatu LAS. Format sestoji iz glave in zajetih točk. V glavi so zajete informacije o posnetih točkah, kot so format točk, mejno polje, odmik točk, merilo, število točk, dolžina zapisa točke in odmik do prve točke [10]. Vsaka točka je predstavljena z lokacijo, intenziteto, klasifikacijsko vrednostjo in ostalimi podatki, kot je vidno v tabeli 1.1.

Tabela 1.1: Primer strukture datotečnega formata LAS.

Podatkovni tip	Lastnost	Dolžina
int[3]	Položaj	12B
unsigned short	Intenziteta	2B
bitfield[3]	Vrednost odboja	3 biti
bitfield[3]	Število odbojev	3 biti
bitfield[1]	Smer zajemanja	1 bit
bitfield[1]	Ali je točka na robu	1 bit
unsigned char	Klasifikacija	1B
unsigned char	Kot zajemanja	1B
unsigned char	Uporabniški podatki	1B
unsigned short	ID izvora točk	2B
double	Čas GPS	8B
unsigned short[3]	Barva točke	6B

Oblak točk običajno prikažemo kot posamezne točke, triangulirano površje ali z uporabo enakomerno razporejene mreže. Z mrežo lahko hitro zgradimo trdno površje iz vhodnega oblaka točk.

2 PREGLED METOD SIMULACIJ TEKOČIN

Simulacija in vizualizacija tekočin v računalništvu je zanimivo področje, ki se uporablja v znanih 3D programskih rešitvah, kot so na primer igralni pogoni. Poznamo različne tipe tekočin (npr. dim, ogenj in voda), ki jih lahko opišemo z enačbami Navier-Strokes (NS). Takšne simulacije temeljijo na kompleksnih enačbah, zato jih moramo najprej diskretizirati, pri čemer imamo na razpolago različne metode. V [13] je opisan pristop diskretizacije enačb NS, kjer enačbe razbijemo v več delov in korakov ter jih rešimo posamezno. Ti deli so običajno advekcija, telesne sile ter pritisk ali nestisljivost tekočin. Splošna oblika NS je definirana kot (2.1, 2.2):

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + v \nabla \cdot \nabla \vec{u}, \tag{2.1}$$

$$\nabla \cdot \vec{u} = 0, \tag{2.2}$$

kjer vektor $\vec{u}$ vsebuje hitrostne komponente (u, v, w), vektor $\vec{g}$ predstavlja gravitacijski pospešek, ρ definira gostoto tekočine in p zračni pritisk. V diskretnem prostoru upoštevamo tudi robne pogoje, ki jih v implementaciji obravnavamo ročno. Ti robni pogoji so na primer stene (robovi mreže), kjer komponentam $\vec{u}$ pravokotno na steno spremenimo predznak. Tekočine lahko predstavimo z mrežo (Eulerjeva metoda), ali s sistemi delcev (Lagrangeova metoda).

Eulerjeva in Lagrangeova metoda v reševanju dinamike tekočin

Ideja Eulerjeve metode je razdeliti površje tekočine v celice. Celice so običajno enakomerno porazdeljene po površju. Vsaka celica vsebuje koordinate (*x, y, z*) in hitrostne komponente (*u, v, w*) v danem trenutku *t* [24]. Položaj celic se ne spreminja, vendar ima vsaka celica drugačne vrednosti skozi čas. Položaj celic je običajno enak položaju celic vhodnega terena, če je ta prisoten.

Z uporabo Lagrangeove metode nismo omejeni na mrežo, ampak lahko tekočine predstavimo s sistemi delcev [7]. Vsak delec vsebuje podobne informacije kot celica v mreži, vendar se s časom spreminja tudi njegov položaj.

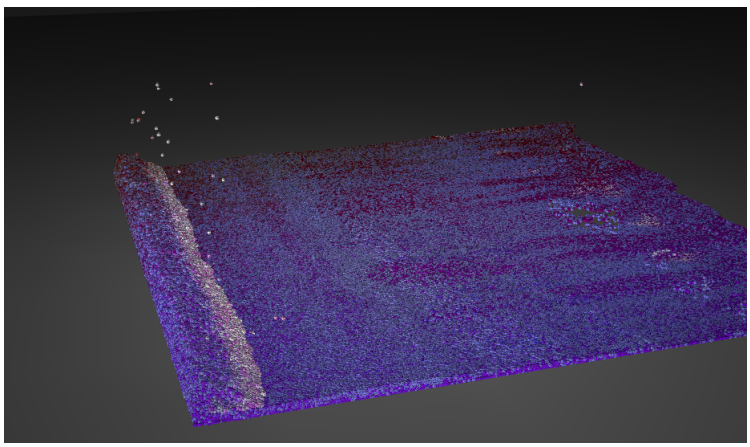
Valovna enačba

Za vizualizacijo enostavnih vodnih površij lahko nad 2D mrežo uporabimo valovno enačbo. Valovna enačba je v osnovi diferencialna enačba, ki opisuje gibanje vsake točke (po višini) na mreži v 2D prostoru. Z implementacijo enačb opisanih v [20], dosežemo hitro in enostavno

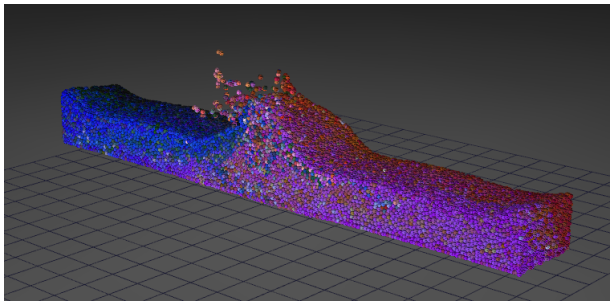
vizualizacijo 2D mreže vodnega površja s spremenljivimi vrednostmi višin celic. Tako je valovna enačba primerna za realno-časovno vizualizacijo oceanov in drugih vodnih površin, ki nimajo stikov s terenom ali kakršnimi koli plavajočimi objekti. Pri metodi lahko spreminjamo koeficiente, kot so viskoznost, hitrost valov, razdaljo in časovni trenutek.

Hidrodinamika zglajenih delcev

Predstavitev vodnega površja z uporabo 2D mrež običajno predstavlja težavo proste gibljivosti celic. Običajno so razdalje med posameznimi celicami konstantne, zato lahko preprosto ustvarimo trikotniško površje nad celotno mrežo. Pri tem se pojavijo težave pri predstavitvi vodnega površja, saj smo zelo omejeni na prostor pri nizkih ločljivostih mreže. Kot alternativo lahko vodno površje predstavimo s sistemom delcev, kjer vsak delec vsebuje različne fizikalne lastnosti, kot so masa, hitrost, položaj in sile [14]. Z implementacijo valovnih enačb nad sistemom delcev lahko dosežemo kompleksne efekte, kot so pljuski trkov valov. Z metodo interakcija delec-delec (ang. particle-particle interaction) temelječe na silah Lennard-Jones, ki nam opisujejo povezavo med posameznimi delci, lahko simuliramo telesa tekočine. Naslednji korak k realističnim simulacijam je metoda hidrodinamike zglajenih delcev (ang. Smothed-Particle Hydrodynamics, SPH), ki so jo razvili predvsem za študije astrofizike [21]. Metoda se je izkazala za robustno in učinkovito ter se je dala aplicirati tudi na probleme ostalih področij, med katere spadajo magnetna hidrodinamika, trdna mehanika in dinamika tekočin [16]. SPH za reševanje dinamike tekočin uporablja enačbe NS z uporabo Lagrangeove metode. Zaradi prostega gibanja delcev je ta metoda primerna za realistične simulacije trkov medsebojnih valov kot tudi trkov z objekti ali terenom. Primer simulacij SPH je mogoč z uporabo orodja Fluids [5] prikazanega na slikah 2.1 in 2.2.



Slika 2.1: Prikaz simulacije SPH s programskim paketom Fluids.



Slika 2.2: Trk dveh valov v programskem paketu Fluids.

Ker je čas izvajanja implementacije SPH zaradi polnih 3D enačb NS na centralno procesni enoti (ang. Central processing unit, CPU) prezahteven za realno-časovno simulacijo, se zadnje čase pojavlja vedno več implementacij na grafičnih procesnih enotah (ang. Graphics processing unit, GPU). Ker lahko rešujemo enačbe NS vzporedno za vsak delec posebej, lahko z malo več truda implementiramo reševalnik enačb z uporabo tehnologije NVIDIA CUDA [8]. Današnji CPU imajo do 6 jeder oziroma do 12 niti, ki so sposobne vzporedno izvajati dane naloge. Vendar ta številka ni niti približno blizu številu jeder današnjih GPU, ki segajo tudi do 1300 jeder. Rustico in soavtorji [23] so predstavili implementacijo SPH z uporabo šetih GPU, kjer je vsaka GPU reševala svoj del simulacije. Na tak način so dosegli hitrejše simulacije z mnogo več pomnilnika za hrambo podatkov na GPU. GPUSHP je odprtokodna aplikacija omenjenih avtorjev, ki omogoča reševanje SPH na GPU [18]. Metoda SPH je eden izmed modelov, ki se lahko uporabi za natančno predvidevanje poplavnih območij nad raznimi oblikami vhodnega terena.

Enačbe plitvih voda

Enačbe plitvih voda (ang. Shallow Water Equations, SWE) oz. enačbe Saint Venant v 1D prostoru [1] so še ena možnost predstavitve gibanja tekočin v prostoru. Uporabne so za različne simulacije, kot so simulacija poplav, tsunamijev, razsutja jezov in močnih neviht [15]. Prav tako so veliko hitrejše in enostavne za implementacijo realno-časovnih simulacij. Te enačbe smo uporabili v implementaciji algoritma v diplomskem delu. Čeprav smo algoritem implementirali na CPU, obstaja tudi veliko implementacij na GPU [15]. Enačbe so primerne tudi za simulacijo oceanov [9], ki temelji na dvoslojnem reševanju z različnim pritiskom tekočine. SWE so običajno implementirane s hibridnim pristopom Eulerjeve in Lagrangeove metode (Semi-Lagrangian method) [13], kjer za predstavitev vode uporabimo enakomerno poravnano mrežo. Pri teh celicah za vsak časovni trenutek izračunamo položaj navideznega delca v času $t - 1$. Za implementacijo SWE lahko uporabimo tudi ostale metode, kot je

na primer metoda Lattice Boltzmann, pri kateri namesto diskretiziranja enačb diskretiziramo model (npr. D2Q9 lattice) [17].

3 IMPLEMENTACIJA ORODJA

Orodje za predvidevanje poplavnih območij smo implementirali v programskem jeziku C/C++, ki omogoča programiranje na nizkem in visokem nivoju. Kot razvojno okolje (ang. Integrated development enviroment, IDE) smo uporabili Visual Studio 2012, ki omogoča urejanje izvorne kode različnih jezikov, kot so C++, C#, HTML, VB, JavaScript. Pri implementaciji orodja smo za doseg ciljev uporabili različne tehnologije, kot sta Windows API [3] in knjižnica DirectX 10.0 [4]. Windows API (Aplication Programming Interface), znan tudi kot Win32 API, je aplikacijski programski vmesnik za pisanje aplikacij Windows. Implementiran je v programskem jeziku C in sestoji iz več funkcij, kot so funkcije za delo z vhodom in izhodom, procesi, nitmi, upravljanjem pomnilnika, funkcijami za ustvarjanje grafičnega uporabniškega vmesnika (GUI - Graphics User Interface), GDI (ang. Graphics Device Interface). Grafični vmesnik našega orodja je ustvarjen z uporabo Windows API. Pri tem v glavnem oknu aplikacije ustvarimo zanko, kjer lovimo sporočila nad oknom in jih posredujemo procedurni funkciji, ki implementira določene dogodke (ang. events) prejetih sporočil. Vmesnik našega orodja smo dopolnili s "trakovi"(ang. Ribbons), ki so podprti od operacijskega sistema (OS) Windows 7 dalje. Ogrodje trakov (ang. Ribbon Framework) temelji na komponentnem objektnem modelu (ang. Component Object Model - COM), prvič pa se pojavi v pisarniškem programskem paketu Microsoft Office 2007. Izgled vmesnika definiramo z označevalnim jezikom XML, ki ga prevedemo v binarno datoteko BML. Dogodki nad kontrolami pa se upravljajo preko vmesnika COM. DirectX 10 je zbirka programskih knjižnic za delo z grafiko, zvokom, vhodom/izhodom in omrežjem. Sestavljena je iz mnogih komponent, med katere spadajo tudi Direct3D, Direct2D, XAudio, XACT, XInput, DirectPlay. Veliko funkcionalnosti iz Direct3D 9 je Direct3D 10 odstranjenih, kot so na primer funkcije SetRenderState, DrawPrimitive. Za izris vsakega gradnika je potreben t.i. program Effect (.FX), v katerem so definirani senčilniki (ang. shaders). Prednost datotek FX je, da si lahko definiramo različne tehnike in prehode, ki se bodo izvajale med izrisom objekta. Te tehnike kličejo različne senčilnike oglišč in pikslov (ang. vertex, pixel shaders). Senčilnik oglišč izvaja operacije nad vsakim ogliščem posebej. Podobno deluje tudi senčilnik pikslov, obstajajo pa še razni drugi senčilniki, kot je na primer senčilnik geometrije, ki kot vhod prejme celoten lik. Če je izbrana topologija izrisa trikotnik, potem senčnik prejme tri oglišča, ki pripadajo temu trikotniku. Na tak način lahko prenesemo izračun normal spreminjajočih se objektov iz CPU na GPU, kjer se izračuni izvajajo vzporedno. Programi FX so napisani v programskem jeziku HLSL (ang. High level shading language), kot prikazuje slika 3.1.

```
struct VertexShaderOutput
{
	float4 Position : SV_POSITION;
	float4 Color  : COLOR0;

};

VertexShaderOutput fVertexShader(float4 pos : POSITION, float4 col : COLOR)
{
	VertexShaderOutput output;
	output.Position = mul(pos, World);
	output.Position = mul(output.Position, View);
	output.Position = mul(output.Position, Projection);
	output.Color = col;
	return output;
}

float4 fPixelShader(VertexShaderOutput input) : SV_TARGET
{
	return input.Color;
}

technique10 Render
{
   pass P0
   {
	SetVertexShader( CompileShader( vs_4_0, fVertexShader() ) );
	SetGeometryShader( NULL );
	SetPixelShader( CompileShader( ps_4_0, fPixelShader() ) );
   }
}
```

Slika 3.1: Vsebina datoteke FX, kjer sta definirana senčilnika oglišč in pikslov.

Grafični pogon

Prvi korak je vzpostavitev grafičnega vmesnika in nato implementacija grafičnega pogona. Najprej vzpostavimo kamero, kjer definiramo koordinatni sistem, matrike in hitrosti premikov. Pri tem preverjamo dogodke vhodnih naprav (tipkovnica - premik po svetu, miška -

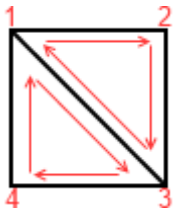
zasuk po oseh X in Y). V pogon implementiramo tudi svoje razrede za hrambo oglišč z uporabo medpomnilnika oglišč (ang. vertex buffer, VB) in indeksov (ang. index buffer, IB). Prav tako potrebujemo datoteke FX za različne tipe oglišč in abstraktni razred *Object*, ki čimbolj poenostavi ustvarjanje geometrije v nadaljevanju implementacije.

3.1 Priprava podatkov

Na začetku implementacije upoštevamo možnost branja višinskih map (ang. height maps). Pri višinskih mapah gre za sliko poljubne dimenzije, kjer položaj piksla predstavlja položaj X in Z v svetu. Vrednost barvnega kanala predstavlja višino Y. Za branje slik uporabimo knjižnico GDI+ (ang. Graphics Device Interface Plus)[6], ki podpira znane slikovne formate, med katere spadajo tudi JFIF (ang. JPEG file interchange format), PNG in BMP. Ponavadi so višinske mape sivinske, zato zadostuje branje samo enega kanala. Ko imamo mrežo pripravljeno, jo shranimo v VB. Ker želimo, da je mreža trdno površje, moramo njene točke povezati v trikotnike. Ker vsako oglišče zasede 40B (položaj, barva in normala) in za vsak trikotnik potrebujemo 3 oglišča, bi to za sliko dimenzij 1000x1000 pomenilo okrog 114 MB pomnilnika v GPU. Ta problem rešimo z IB, katerega napolnimo z informacijami o povezavah med oglišči. Na tak način zmanjšamo zasedenost površja v pomnilniku. Velikost VB izračunamo kot zmnožek višine in širine višinske mape (3.1):

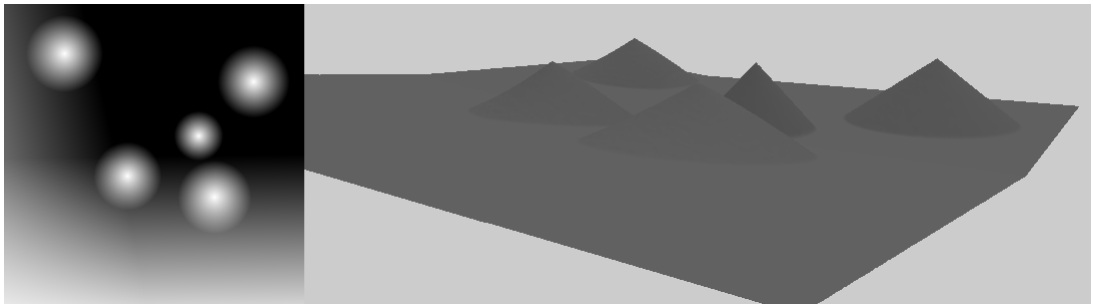
$$velikost_{vb} = visina \cdot sirina. \tag{3.1}$$

Vsaka celica sestoji iz dveh trikotnikov, za katera moramo povedati, katerim ogliščem pripadata (slika 3.2).



Slika 3.2: Orientacija povezovanja ogjišč v trikotnike. V IB se za celico na sliki zapiše 6 vrednosti (1, 2, 3, 3, 4, 1) za predstavitev obeh trikotnikov.

Primer višinske mape in zgrajen 3D model je viden na sliki 3.3.



Slika 3.3: Višinska mapa (levo) in njen 3D model (desno).

Velikost IB potrebnega za povezovanje oglišč izračunamo z (3.2):

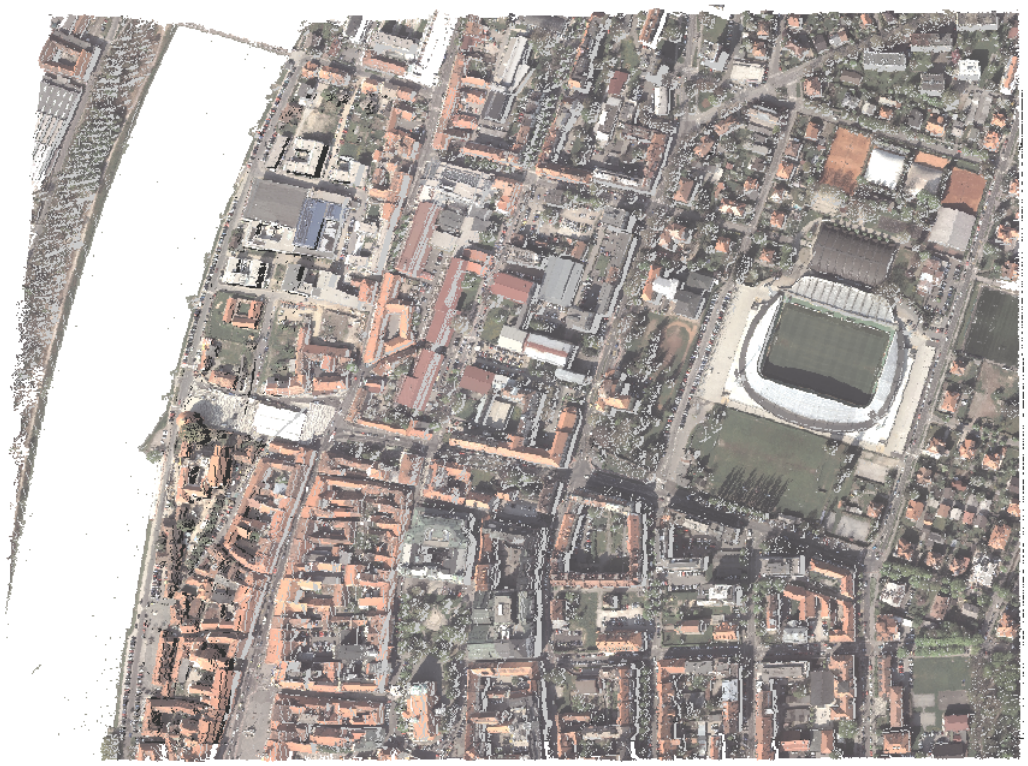
$$velikost_{ib} = (visina - 1) \cdot (sirina - 1) \cdot 6. \tag{3.2}$$

3.2 Branje podatkov LiDAR

V orodju prav tako podpiramo možnost branja datotek LAS (LASer File Format). Najprej iz glave datoteke preberemo število vseh točk, alociramo dovolj velik pomnilnik in vanj preberemo vse točke. Točke v LAS niso enakomerno razporejene (glej sliko 3.4). V GUI imamo na voljo uporabniško nastavljivo ločljivost vhodne datoteke LAS, ki zgradi mrežo po dani ločljivosti. Širina in višina mreže se izračunata z (3.3) in (3.4):

$$sirina = \frac{(max_x - min_x)}{loljivost} + 1, \tag{3.3}$$

$$visina = \frac{(max_y - min_y)}{loljivost} + 1. \tag{3.4}$$



Slika 3.4: Prikaz oblaka točk LiDAR dela mesta Maribor.

Točke iz LAS so georeferencirane in jih moramo najprej pretvoriti v koordinate mreže od (0,0) do (širina,višina) in preveriti, v katero celico pade. Višina celice je enaka višini zadnje padle točke v to celico. Koordinate izračunamo po enačbah (3.5, 3.6 in 3.7), pri čemer vrednosti *x* in *y* zaokrožimo navzdol.

$$x = \lfloor \frac{(x'scale_x - min_x) + offset_x}{loljivost} \rfloor \tag{3.5}$$

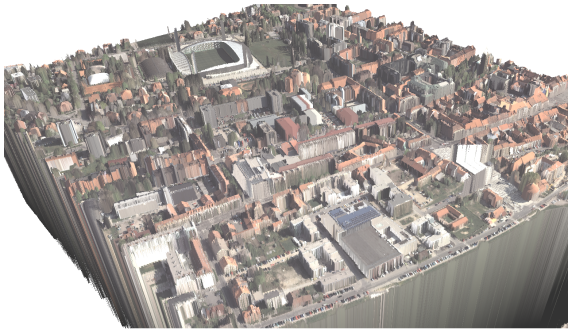
$$y = \lfloor \frac{(y'scale_y - min_y) + offset_y}{loljivost} \rfloor \tag{3.6}$$

Višina celice izračunamo kot:

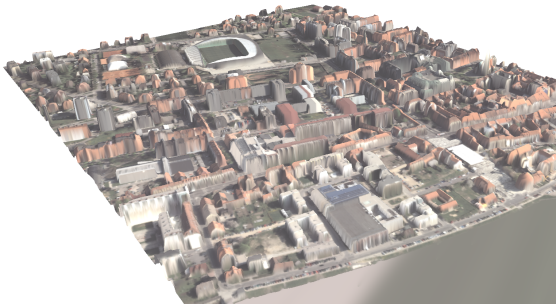
$$z = \frac{z'scale_z - min_z}{loljivost} + 1. \tag{3.7}$$

Ker naše orodje uporablja drugačen koordinatni sistem kot datoteka LAS, moramo dobljene koordinate pretvoriti. Za vsako celico si hranimo vrednosti klasifikacije, barve in intenzitete točke. Ko končamo z gradnjo mreže (glej sliko 3.5), je potrebno popraviti še teren. Celice označene kot teren ali stavbe ohranimo, višine ostalih celic pa povprečimo z vrednostjo sosednjih celic, ki so klasificirane kot teren ali stavba. Ta korak rekurzivno ponavljamo, dokler

niso vse celice na nivoju terena ali stavb, kot je vidno na sliki 3.6. Iz pridobljene mreže zgradimo objekte VB in IB enako kot pri nalaganju višinskih map.



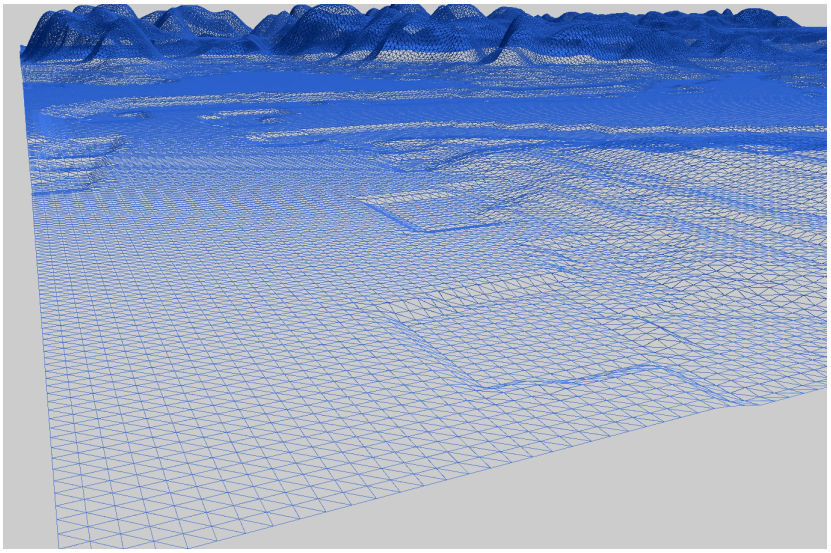
Slika 3.5: Originalen LiDAR.



Slika 3.6: Očiščen LiDAR, kjer je odstranjen šum in vegetacija ter so zapolnjene luknje v površju.

3.3 Gradnja sloja tekočin

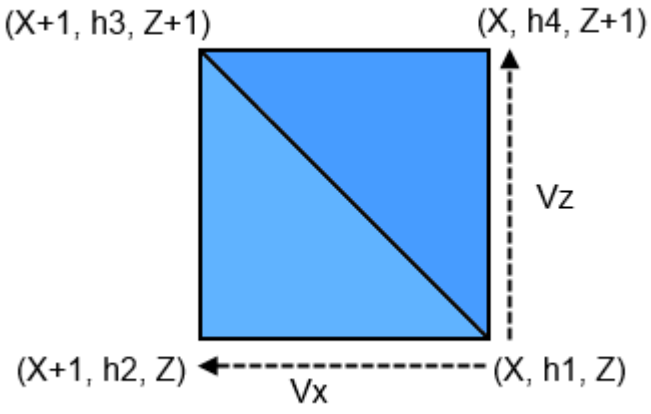
Po nalaganju vhodnih podatkov ustvarimo še vodno površje nad danim terenom. Vsaka vodna celica ima malo drugačno ogliščno deklaracijo (podatki o oglišču), saj še zraven položaja, barve in normale hranimo vrednost ali je celica izbrana. Celica sestoji iz spremenljivk, kot so položaj, hitrost, višina terena, višina vode, globina vode, minimalna višina terena, maksimalna višina terena, maksimalna globina vode, faktor polnjenja in stanje celice. Vse te vrednosti nastavimo pri inicializaciji vodnega površja. Položaj (x, y) vodne celice je enak položaju celice na terenu (x, z). Komponente hitrostnega vektorja $\vec{v}$ nastavimo na 0, višina terena h je enaka koordinati Y celice terena. Prav tako sta višina in globina vodnega površja manjša od višine terena in vse celice imajo stanje nastavljeno na suho, kar pomeni da ni poplavljenih predelov. Za vodno površje prikazano na sliki 3.7, potrebujemo nov objekt VB in IB.



Slika 3.7: Mreža vodnega površja.

3.4 Algoritem

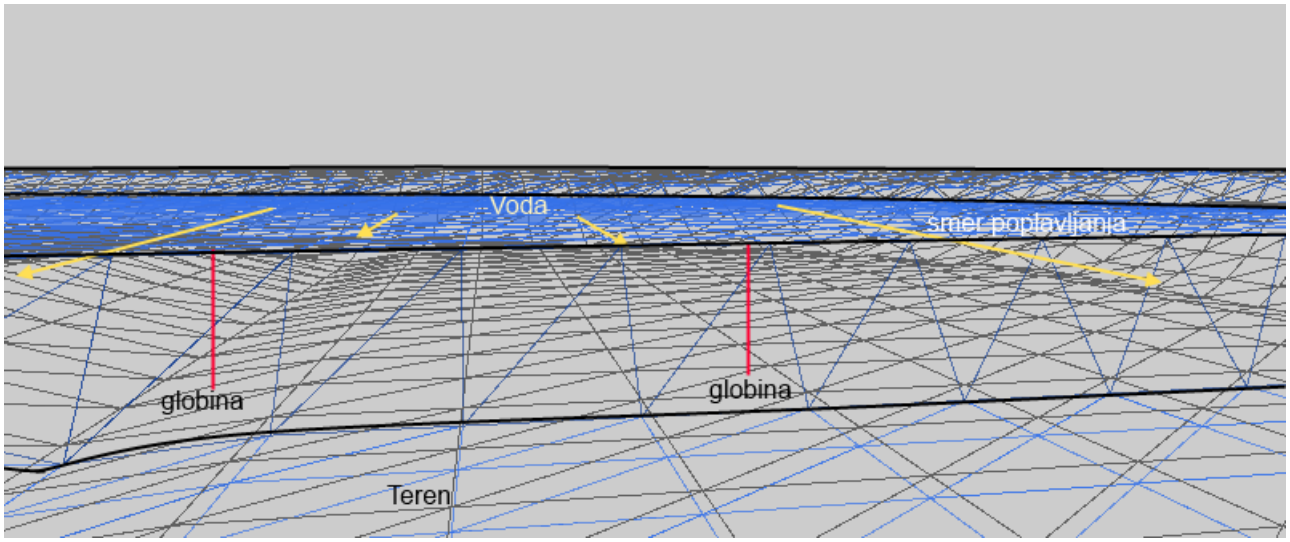
V tem diplomskem delu predstavimo algoritem predvidevanja poplavnih območij z uporabo SWE, ki so poenostavljene oblike enačb NS in opisujejo gibanje tekočin v 2D prostoru z dodatnim poljem globine [25]. Algoritem deluje nad 2.5D mrežo, ki jo zgradimo med inicializacijo vodnega sloja. Vsaka točka (slika 3.8) v mreži vsebuje lastnosti, kot sta položaj (x, y) in višina h.



Slika 3.8: Vodna celica s štirimi točkami.

Vsaka točka ima enake koordinate kot točka terena. Hitrost točke (v_x, v_y) nam pove, v katero

smer, oz. s kakšno hitrostjo se ta premika (prenese vrednost globine na ostale točke v njeni smeri) po vodni gladini, kar je prikazano na sliki 3.4.



Slika 3.9: Mreža terenskega in vodnega sloja.

Prostornino poplavljenega terena izračunamo z 3.8:

$$V = \sum_{i=0}^{n} (globina_i \cdot loljivost^2). \tag{3.8}$$

POSTOPEK ALGORITMA:

1. Najprej ustvarimo začasno kopijo vrednosti mreže vodnega površja, ki jo potrebujemo za izračun novih vrednosti.

2. Naslednji korak algoritma je advekcija višin in hitrosti celic. Najprej izračunamo povprečne vrednosti hitrosti (v_x, v_y) sosednjih celic. Z uporabo povprečne hitrosti lahko določimo položaj (3.9, 3.10) kje se je ta celica nahajala v času $t-1$:

$$x' = x - \Delta t v_x, \tag{3.9}$$

$$y' = y - \Delta t v_y. \tag{3.10}$$

Ker je nov položaj točke lahko kjerkoli na terenu, moramo najprej ugotoviti v kateri celici se ta točka nahaja, kar je trivialno z uporabo mreže. Ker poznamo položaje (x, y) in globine točk h v celici ter položaj nove točke (x', y'), lahko z bilinearno interpolacijo (3.11, 3.12) izračunamo njeno globino h'(3.13):

$$r_1 = \frac{x_{10} - x'}{x_{10} - x_{00}} h_{00} + \frac{x' - x_{00}}{x_{10} - x_{00}} h_{10}, \tag{3.11}$$

$$r_2 = \frac{x_{10} - x'}{x_{10} - x_{00}} h_{01} + \frac{x' - x_{00}}{x_{10} - x_{00}} h_{11}, \tag{3.12}$$

$$h' = \frac{y_{11} - y'}{y_{11} - y_{10}} r_1 + \frac{y' - y_{10}}{y_{11} - y_{10}} r_2. \tag{3.13}$$

Točki priredimo interpolirano vrednost h'. Enako storimo tudi pri advekciji hitrosti v_x in v_y, pri čemer namesto vrednosti celic h v enačbo vstavimo v_x in v_y.

3. Nato posodobimo še vse globine $h_{x,y}$ v mreži. Najprej si znova ustvarimo kopijo mreže in izračunamo novo vrednost h', ki lahko predstavlja naraščanje ali ponikanje vode z (3.14):

$$h'_{i,j} = h_{i,j}((\frac{v^x_{i-1,j} - v^x_{i,j}}{d} + \frac{v^y_{i,j+1} - v^y_{i,j}}{d}) * \Delta t), \tag{3.14}$$

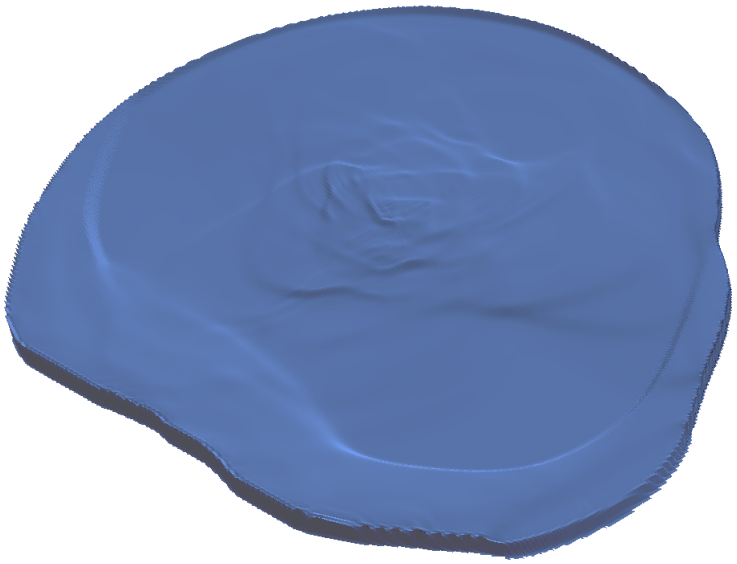
kjer d predstavlja velikost celice (v našem primeru vedno 1) in je Δt časovni korak. Po končani posodobitvi globin, izračunamo še višine točk μ, ki so vsota višine terenskih in vodnih točk.

4. Po izračunu novih višin, izračunamo nove vrednosti hitrosti. Hitrostna koeficienta $v^x_{i,j}$ in $v^y_{i,j}$ posodobimo le v primeru že poplavljene celice, sicer bi se posodobile tudi hitrosti suhih celic, kar pa bi slabo vplivalo na premike ostalih celic v mreži. Nove hitrosti izračunamo v dveh korakih. Najprej izračunamo nove vrednosti za $v^x_{i,j}$ (3.15), kjer je $i \geq 0, i < sirina_x - 2$ in $j \geq 1$, $j < visina_z$. Nato še posodobimo vrednosti $v^y_{i,j}$ (3.16), kjer je $i \geq 0$, $i < sirina_x - 1$ in $j \geq 2$, $j < visina_z$. Enačbi za vsak element sta naslednji:

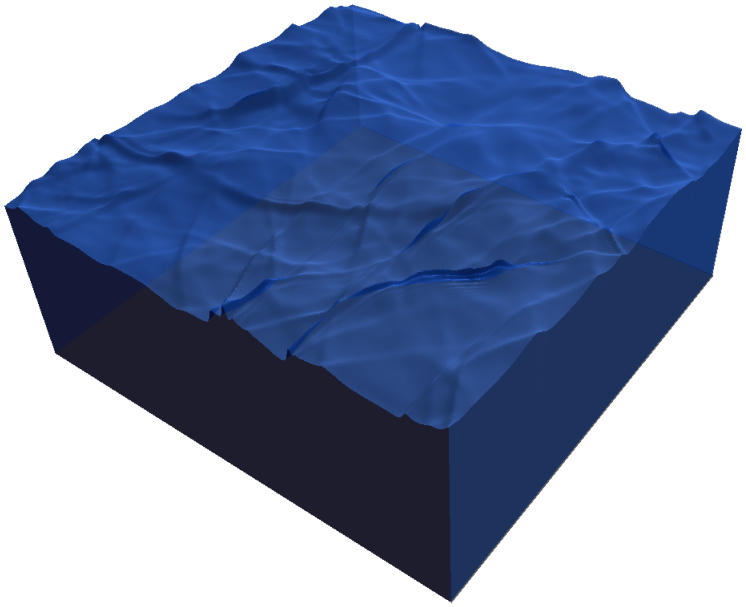
$$v^x_{i,j} = v^x_{i,j} + g(\mu_{i+1,j} - \mu_{i,j})\Delta t, i \in \{0, 1, ..., sirina_x - 3\}, j \in \{1, 2, ..., visina_z - 1\}, \quad (3.15)$$

$$v^y_{i,j} = v^y_{i,j} + g(\mu_{i,j-1} - \mu_{i,j})\Delta t, i \in \{0, 1, ..., sirina_x - 2\}, j \in \{2, 2, ..., visina_z - 1\}. \quad (3.16)$$

V osnovi je za simulacijo vodne dinamike to dovolj (glej sliko 3.10). Za boljše delovanje upoštevamo tudi robne pogoje, s katerimi popravimo vrednosti na robovih, kar nam omogoča odbijanje toka od navideznih sten okoli mreže (slika 3.11).



Slika 3.10: Primer sloja tekočin osnovne metode SWE.



Slika 3.11: Primer sloja tekočin skupaj z odboji tokov od sten.

5. Ker generiran teren ni ravna ploskev ampak je lahko precej hribovit, je potrebno upoštevati dodatna stanja in enačbe dodatnega prilivanja tekočine. Uporabimo metodo prostih površjih [25], kjer za vsako točko poiščemo vrednosti najnižje in najvišje točke v okolici (sosednje točke) in določimo povprečno najnižjo točko terena h_{min} z (3.17):

$$h_{i,j}^{min} = \frac{h_{i,j}^{teren} + min(h_{i,j}^{sosedi})}{2}. \tag{3.17}$$

Določimo tudi povprečno najvišjo točko terena z (3.18):

$$h_{i,j}^{max} = \frac{h_{i,j}^{teren} + max(h_{i,j}^{sosedi})}{2}) + \theta, \tag{3.18}$$

kjer θ predstavlja majhno vrednost, da zagotovimo neenakost med $h_{i,j}^{min}$ in $h_{i,j}^{max}$. To vrednost si predstavljamo kot prag ali napetost površja, s katerim določimo, ali bo celica poplavljena. Poleg najvišje in najnižje točke terena določimo še najvišjo točko vodnega površja $\mu_{i,j}^{max}$ z (3.19) po podobni enačbi kot pri 3.17:

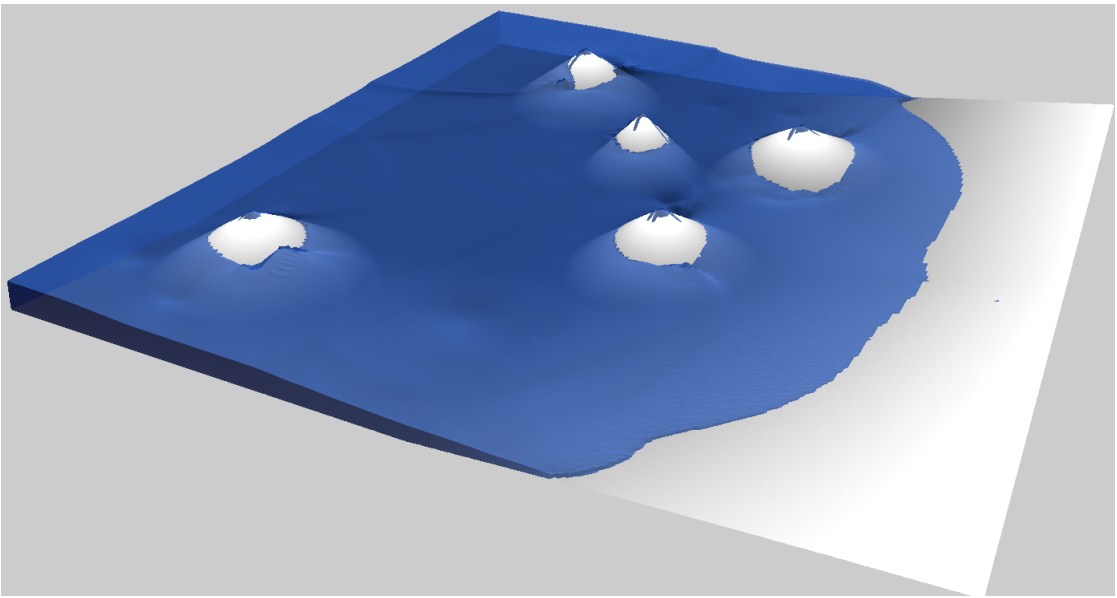
$$\mu_{i,j}^{max} = \frac{\mu_{i,j} + max(\mu_{i,j}^{sosedi})}{2}. \tag{3.19}$$

Nato določimo še stanje vsake točke. Kadar je višina točke terena manjša ali enaka povprečni vrednosti trenutne in najmanjše višine sosednjih točk ($h_{i,j}^{teren} \leq h_{i,j}^{min}$) ter so vse višine vode sosednjih točk z določenim pragom večje od povprečne vrednosti trenutne in najvišje sosednje točke ($\mu_{i,j}^{max} < (\mu_{i,j}^{sosedi}) + \theta$), takrat točko (i,j) obravnavamo kot suho. Točki nastavimo

faktor polnjenja (ang. fill) na 0. Točka je poplavljena, kadar je višja od terenskih točk v celici. Takrat nastavimo faktor polnjenja na 1. Faktor polnjenja nam omogoča naraščanje globine vode točke glede na sosednje točke. Če imajo sosednje točke višjo globino vode, izravnamo to globino z globino trenutne točke. Faktor polnjenja uporabimo pri izračunu trenutne globine točke.
Kadar noben pogoj ni izpolnjen je globina dovolj narasla in začnemo postopoma zmanjševati faktor polnjenja po (3.20) do najnižje dovoljene vrednosti, ki je uporabniško nastavljiva.

$$f_{i,j} = \frac{h_{i,j}^{teren} - h_{i,j}^{min}}{h_{i,j}^{max} - h_{i,j}^{min}}. \tag{3.20}$$

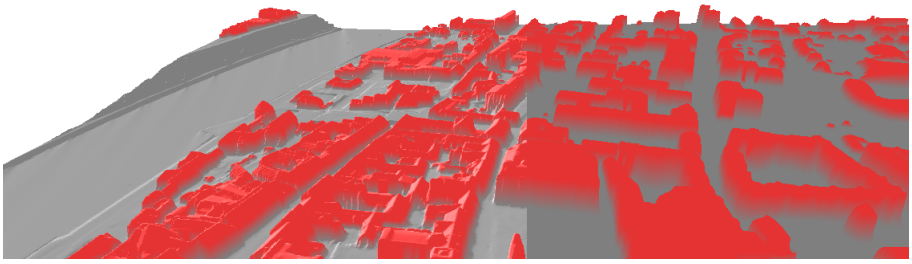
Sedaj SWE deluje tudi nad hribovitimi površji (slika 3.4).



Slika 3.12: Primer prilagajanja tekočine terenu.

3.5 Vizualizacija

Vsaka točka terena je predstavljena s položajem, barvo in normalo. Tem točkam izračunamo normale in jih naložimo v VB. Pri tem VB razpolaga z grafičnim pomnilnikom, zato lahko točke odstranimo iz sistemskega pomnilnika. Normale uporabljamo za senčenje površja za lažjo prepoznavnost objektov in reliefa (glej sliko 3.13).



Slika 3.13: Vizualizacija s senčenjem (levo) in brez senčenja (desno).

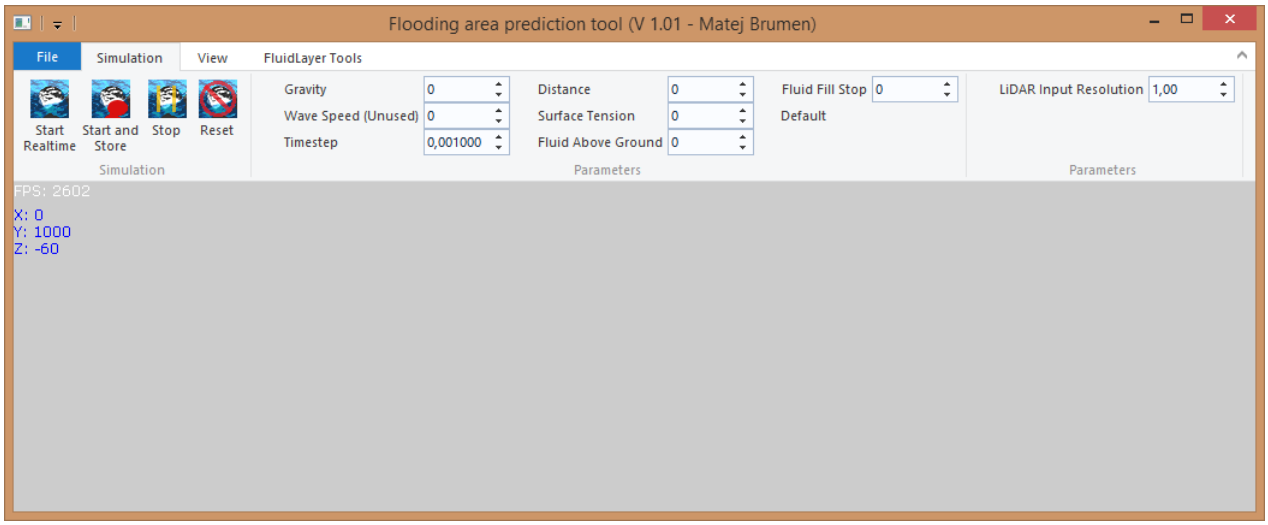
Ker vemo, da se oblika terena ne spreminja, je normala na vsako celico vedno enaka. V primeru vodnega površja, ki se spreminja s časom, moramo za vsako iteracijo izračunati nove vrednosti normal. Pri tem se pojavi težava, saj je računanje normal nad celotno mrežo zelo potratno na CPU. Ker pa Direct3D 10 omogoča uporabo senčilnika geometrije, je ta primeren za računanje normal na GPU.

Naše orodje omogoča označevanje vodnih celic, zato dodamo vsakemu oglišču spremenljivko, ki nam pove ali je točka označena. Ta vrednost potuje čez cevovod D3D10 do senčilnika pikslov, ki v primeru, če je ta vrednost enaka 1, pobarva vse piksle tega oglišča z izbrano barvo. S tem seveda ne posegamo v začetne barve točk dodeljene med inicializacijo vodnega površja. Vse senčilnike hranimo v datoteki FX, ki jo naložimo za vsak objekt, ki ga ustvarimo. FX nam omogoča uporabo različnih tehnik upodabljanja. Na primer, za upodabljanje terena uporabimo drugačne senčilnike oglišč in pikslov kot pri upodabljanju vodnega površja, ki uporablja še senčilnik geometrije.

Pri upodabljanju vodnega površja smo poleg senčenja upoštevali še globino posamezne točke, ki jo prenesemo vse do senčilnika pikslov, kjer potemnimo vrednost pikslov glede na vrednost globine.

4 REZULTATI

Po končani implementaciji aplikacije smo se osredotočili na njeno uporabo. Uporabniški vmesnik in interakcija orodja sta uporabniško prijazna, kamero upravljamo z miško in tipkovnico, kar nam omogoča pogled terena iz različnih lokacij (glej sliko 4.1).



Slika 4.1: Uporabniški vmesnik našega orodja.

Orodje omogoča naslednje funkcionalnosti:

- nalaganje višinskih map,
- nalaganje podatkov LiDAR z uporabniško določljivim parametrom ločljivosti mreže,
- uporabniške kontrole za začetek, pavzo in ponastavitev simulacije,
- spreminjanje parametrov algoritma med njegovim delovanjem,
- topologija prikaza terena in vodnega površja (Omogočeno je prikazovanje točk, mreže in trdnega površja. Uporabniku je določeno tudi skrivanje slojev.),
- uporabniško označevanje posameznih celic in nastavljanje njihovih višin.

Parametri algoritma, ki se upoštevajo, so:

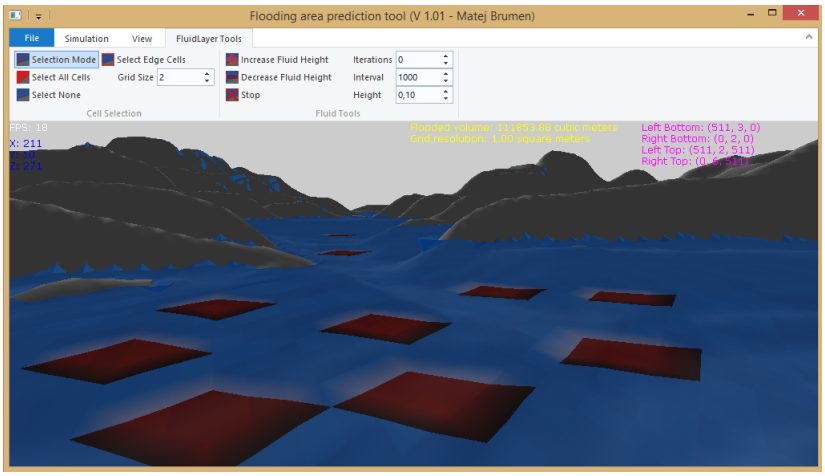
1. Gravitacijska sila g vpliva na hitrost premika celic v enačbah (3.15) in (3.16). S spreminjanjem te vrednosti nadziramo spremembo hitrosti dveh celic različnih višin. Višja je razlika v višinah in višja je sila gravitacije, večja bo hitrost toka celice v smeri x in y.

2. Časovni korak Δt nam omogoča spreminjanje hitrosti izvajanja simulacije. Vpliv parametra opisujejo enačbe 3.9, 3.10, 3.14, 3.15 in 3.16. Višja je vrednost, večje bodo razdalje med novimi koordinatami točke v času $t-1$, večja bo stopnja pojemanja vode. Hitrost toka je tudi odvisna od te vrednosti. Pri spreminjanju vrednosti pazimo na stabilnost algoritma, saj se z višanjem vrednosti ta zmanjšuje, zato je pri višjih globinah priporočljivo zmanjšati časovni korak.

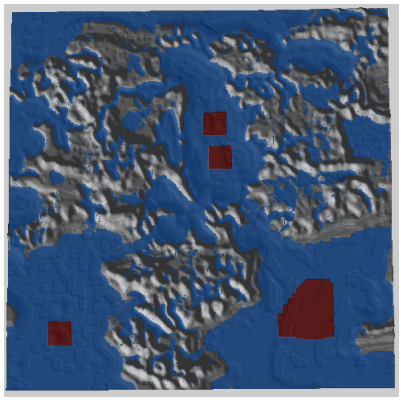
3. Napetost površja Θ se uporablja za simulacijo različnih materialov površja. Na primer voda lažje teče po gladkih površinah. Vrednost se uporablja v enačbi (3.18).

4. Minimalna dovoljena višina vode je vrednost, do katere je lahko celica poplavljena. Ko globina celice pade pod to vrednost, se hitrost, globina in stanje celice izničijo.

Uporabniku orodja je prav tako omogočeno označevanje posameznih celic na mreži (slika 4.2). V GUI lahko aktiviramo označevalni način, ki omogoča označevanje celic s klikom na vodno površje. Zraven imamo še dodatne gumbe, ki nam olajšajo označevanje, saj omogočajo označevanje vseh celic (tudi robnih) in odznačevanje celic. V orodju lahko nastavimo tudi površino označevanja celic. Tako lahko izbiramo večja območja in nad njimi izvajamo implementirane operacije, kot je vidno na sliki 4.3.

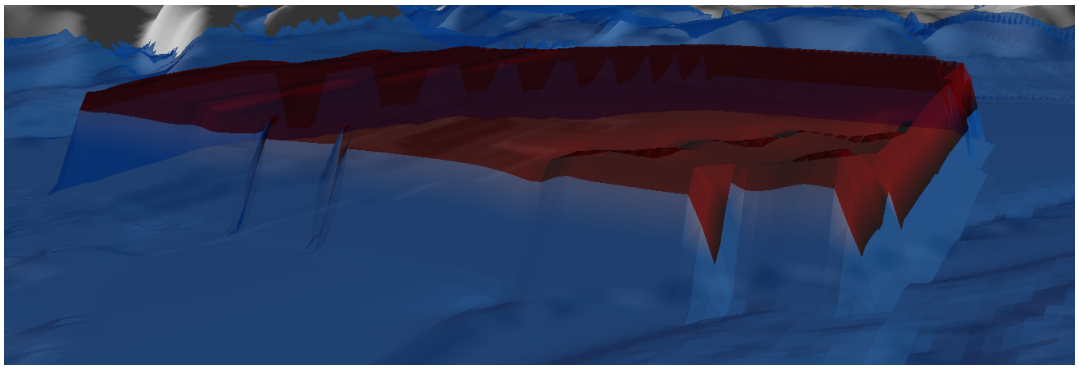


Slika 4.2: Grafični vmesnik za manipulacijo vodnih celic.



Slika 4.3: Upodabljanje posameznih izbranih celic.

Označenim celicam lahko tudi spreminjamo višino (slika 4.4), jim določimo časovni interval in število iteracij naraščanja. To nam omogoča avtomatizirano naraščanje količine vode v določenih predelih v določenem trenutku.



Slika 4.4: Upodabljanje dviga globine izbranih celic.

4.1 Eksperimenti

Algoritem smo preizkusili na različnih površjih LiDAR, kjer smo na znanih lokacijah rek, jezer in potokov ali kar na terenu označili vodna površja. Označenim delom smo nastavili višino in interval naraščanja vode (na primer: tekočina označenih predelov naraste 10cm vsakih 5 sekund). Zaradi zahtevnosti algoritma smo morali zmanjšati ločljivost vhodnih podatkov, čeprav smo lahko aplikacijo nemoteno uporabljali do meje 300000 naloženih točk. Po tej vrednosti je aplikacija težje uporabna za pogon reševalnika SWE.

Strojna oprema, na kateri smo testirali delovanje programskega paketa, ki je bila:

- CPU: Intel Core i7 (3.60GHz)

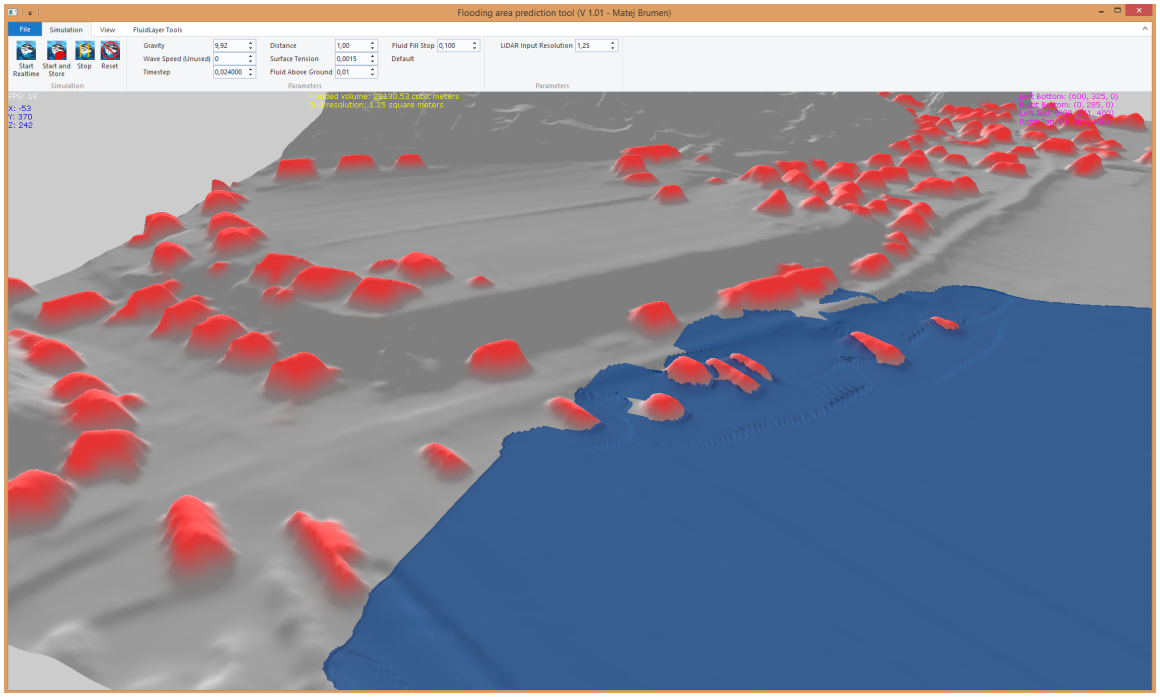
- Pomnilnik: 16GB DDR3
- Grafična enota: NVIDIA GeForce GTX 660 (2GB)

Z naraščanjem števila točk smo opazili močno upadanje zmogljivosti algoritma, ki je tekel na enem jedru CPU. Primerjava je prikazana v tabeli 4.1. Aplikacija je delovala realno-časovno, dokler smo naložili teren z manj kot 100000 točkami.

Tabela 4.1: V tabeli je primerjava števila slik na sekundo (FPS) za različno število točk.

		FPS		
#	Število točk	Teren	Teren in voda	skupaj z algoritmom
1	46750	1151	1073	660
2	83250	1050	880	272
3	187500	1150	425	22
4	333000	1106	246	13
5	750000	648	115	6
6	3000000	160	29	1
7	4056544	139	18	< 1
8	6760000	77	10	< 1

Algoritem je deloval po naših pričakovanjih. Najprej se je voda razširila po nižjih predelih. Vodni tok celic se je spremenil, kadar so bile na poti ovire (npr. stavbe ali visok teren). Kadar je bil vodni tok dovolj visok, je voda prodrla na višji teren ali poplavila določene stavbe. Algoritem smo testirali tudi z različnimi nastavljenimi parametri, pri čemer smo morali paziti na stabilnost algoritma.
Na sliki 4.5 vidimo, kako se vodni tok ob trku s stavbami razcepi na več delov.



Slika 4.5: Vizualizacija trka vode s stavbami (rdeči objekti) in posledično spremembo vodnega toka.

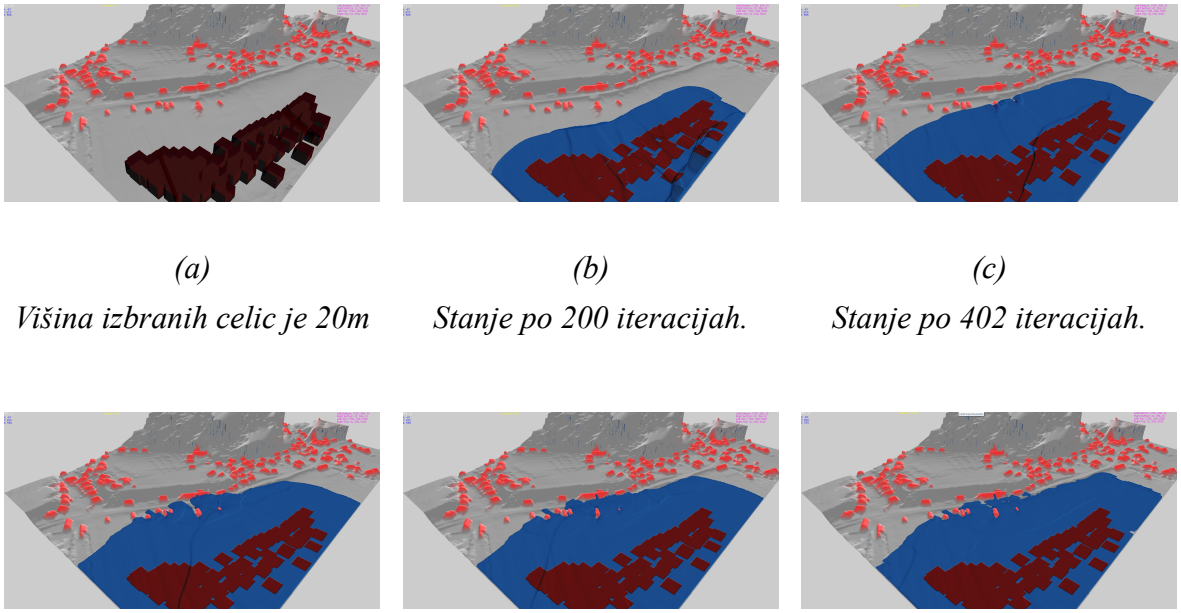
Naredili smo nekaj različnih poizkusov poplavljanja nad različnimi tereni. Štiri izmed teh predstavimo v nadaljevanju.

Poizkus 1

Tabela 4.2: Lastnosti terena pri poizkusu št. 1

Ime datoteke	"H231207.las"
Gravitacija	9.92
Časovni korak	0.039
Ločljivost terena	1 točka na m^2
Začetna višina izbranih celic	izbranih celic: 20m
Število vodnih celic	372000
Velikost mreže	750 x 500

Lastnosti terena pri poizkusu 1 so vidne v tabeli 4.2. Na prvem preizkušenem terenu smo si zamislili območje, ki bo izvor poplav. Celice smo dvignili za 20m, sprožili simulacijo in opazovali stanje poplav vsakih 200 iteracij. Rdeči objekti na sliki 4.6 predstavljajo stavbe.



(a) *Višina izbranih celic je 20m*

(b) *Stanje po 200 iteracijah.*

(c) *Stanje po 402 iteracijah.*

(d) *Stanje po 600 iteracijah.*

(e) *Stanje po 800 iteracijah.*

(f) *Stanje po 998 iteracijah.*

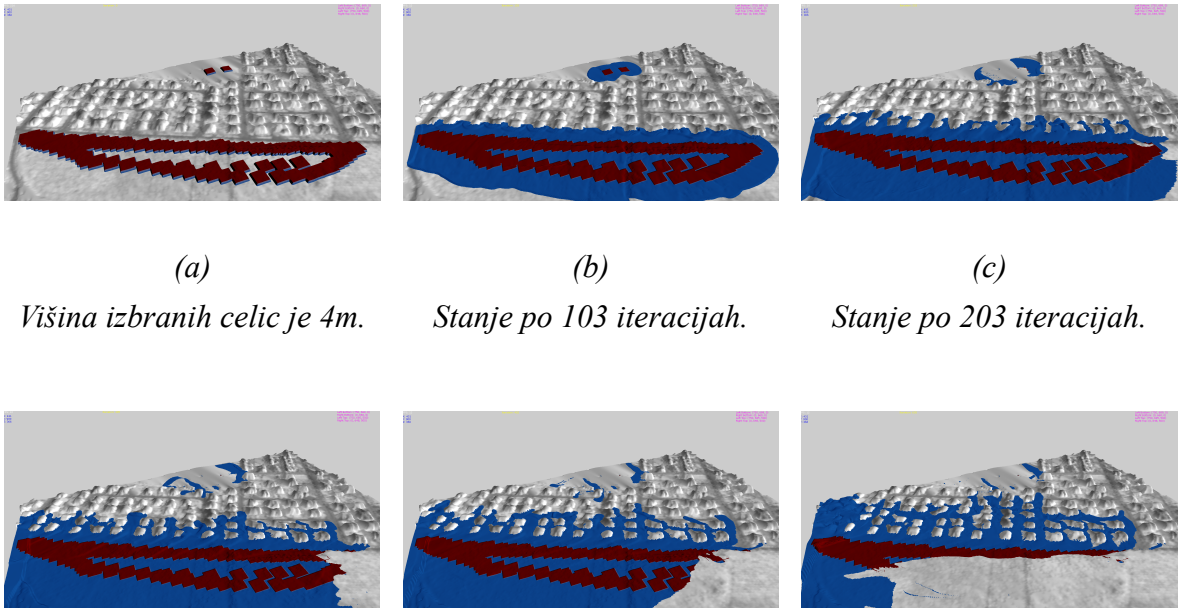
Slika 4.6: Poizkus št 1.

Poizkus 2

Tabela 4.3: Lastnosti terena pri poizkusu št. 2

Ime datoteke	"I271110.las"
Gravitacija	9.92
Časovni korak	0.07
Ločljivost terena	1 točka na m^2
Začetna višina izbranih celic	izbranih celic: 4m
Število vodnih celic	513872
Velikost mreže	750 x 689

Lastnosti terena pri poizkusu 2 so vidne v tabeli 4.3. Vzporedno z vasjo smo označili celice in jih dvignili za 4m. Sprožili smo simulacijo in opazovali, kako se je voda razlila med ulice vasi. Po okoli 600 iteracijah je začela voda pojemati, kot je vidno na sliki 4.7.



(a) *Višina izbranih celic je 4m.*

(b) *Stanje po 103 iteracijah.*

(c) *Stanje po 203 iteracijah.*

(d) *Stanje po 305 iteracijah.*

(e) *Stanje po 402 iteracijah.*

(f) *Stanje po 605 iteracijah.*

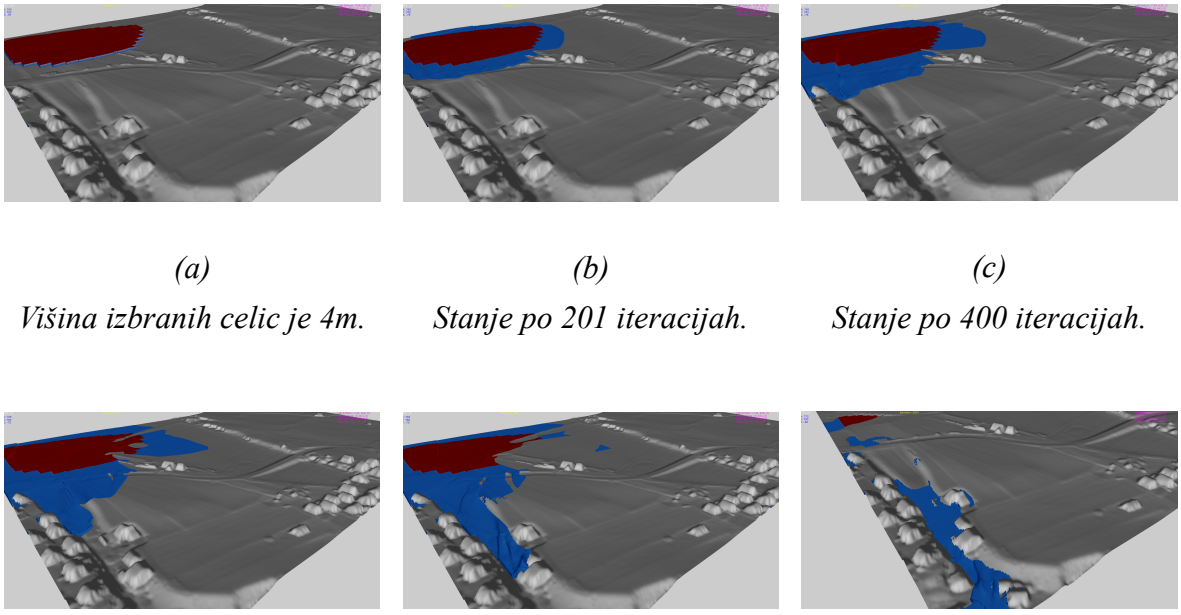
Slika 4.7: Poizkus št 2.

Poizkus 3

Tabela 4.4: Lastnosti terena pri poizkusu št. 3

Ime datoteke	"D241017.las"
Gravitacija	9.92
Časovni korak	0.059
Ločljivost terena	1 točka na m^2
Začetna višina izbranih celic	izbranih celic: 4m
Število vodnih celic	467618
Velikost mreže	749 x 628

Lastnosti terena pri poizkusu 3 so vidne v tabeli 4.4. Na površju (slika 4.8) smo označili celice in jim nastavili višino na 4m. Tekočina celic se je hitro razlila po dokaj ravnem površju. Preostala tekočina je hitro za tem poplavila majhno dolino.



(a) *Višina izbranih celic je 4m.*

(b) *Stanje po 201 iteracijah.*

(c) *Stanje po 400 iteracijah.*

(d) *Stanje po 601 iteracijah.*

(e) *Stanje po 802 iteracijah.*

(f) *Stanje po 2504 iteracijah.*

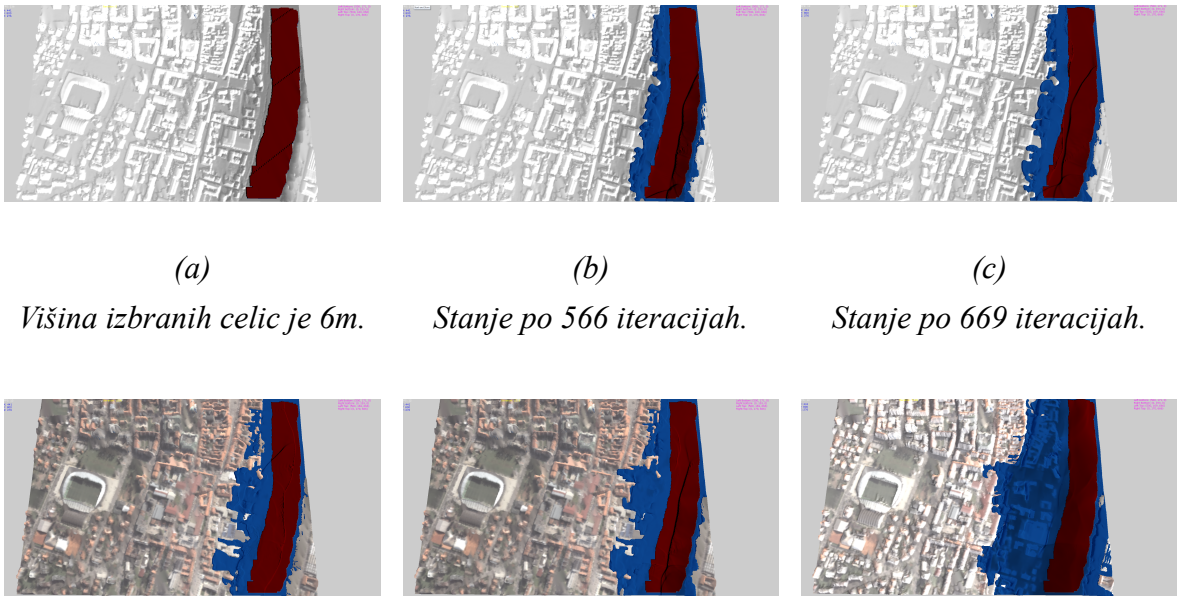
Slika 4.8: Poizkus št 3.

Poizkus 4

Tabela 4.5: Lastnosti terena pri poizkusu št. 4

Ime datoteke	"maribor.las"
Gravitacija	10
Časovni korak	0.059
Ločljivost terena	1 točka na $1,5m^2$
Začetna višina izbranih celic	izbranih celic: 6m (se spreminja s časom)
Število vodnih celic	330668
Velikost mreže	500 x 666

Lastnosti terena pri poizkusu 4 so vidne v tabeli 4.5. Metodo smo preizkusili tudi nad podatki LiDAR mesta Maribor. Najprej smo ročno označili predel reke Drave, nastavili parametre in spremljali poplavljanje nižjih predelov. Na naslednjih slikah *4.9a, b, c* so zaradi lažje predstavitve višine terena celice pobarvane glede na višino le tega. Višino izbranih celic smo spreminjali s časom. Celice terena so na slikah *4.9d, e, f* pobarvane glede originalnih barv terena, pridobljenega iz zračnega posnetka ortofoto.



(a) *Višina izbranih celic je 6m.*

(b) *Stanje po 566 iteracijah.*

(c) *Stanje po 669 iteracijah.*

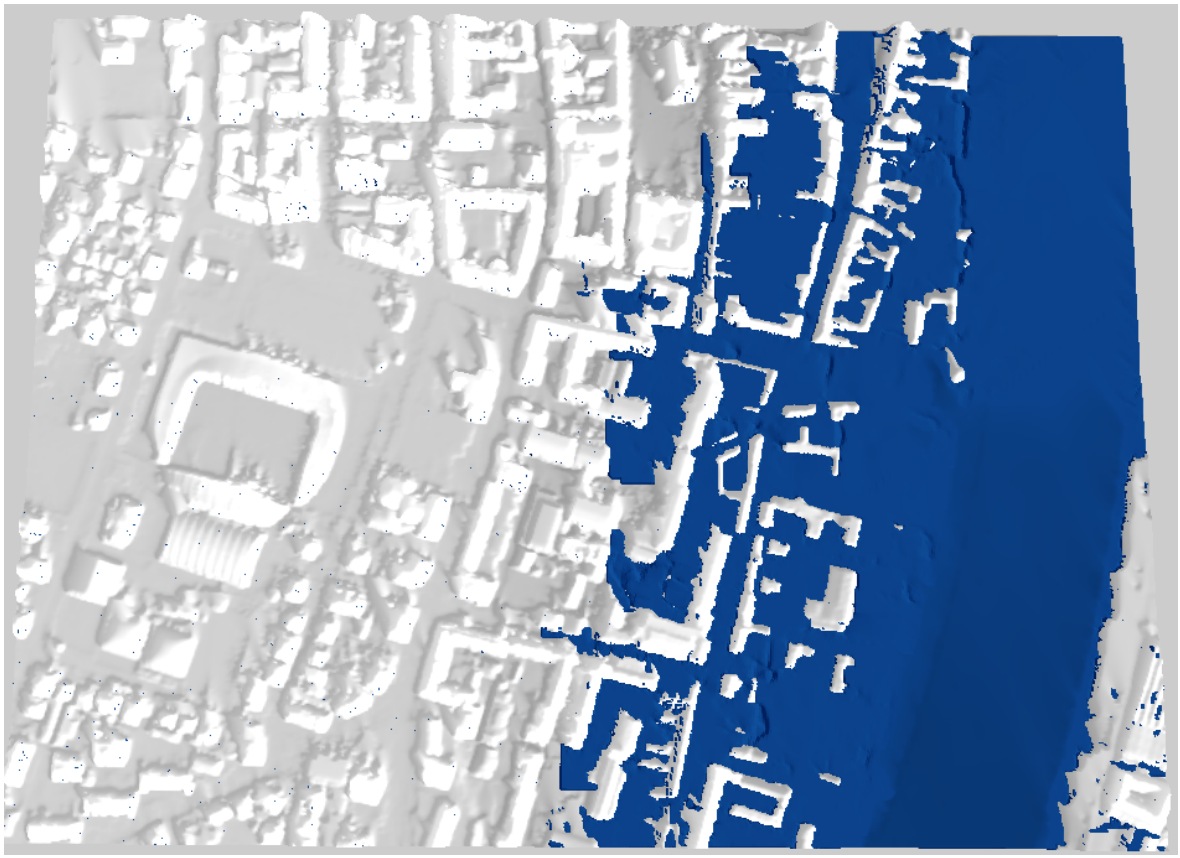
(d) *Stanje po 1000 iteracijah.*

(e) *Stanje po 1519 iteracijah.*

(f) *Stanje po 5189 iteracijah.*

Slika 4.9: Poizkus št 4.

Na zadnji sliki (4.1) smo prikazali umirjeno stanje algoritma, kjer se sloj vode spreminja zelo počasi.



Slika 4.10: Stanje po 10000 iteracijah algoritma (umirjeno stanje).

Ker si lahko z našim orodjem izmislimo različne scenarije poplavljanja, smo pri določenih terenih pretiravali z naraščanjem vode. Kot rezultat smo na določena območja izlili tudi preko 13000 hl tekočine. S tem smo ugotovili, da je algoritem primeren tudi za predvidevanje poplavnih območij primorskih mest ali pokrajin.

5 SKLEP

V diplomskem delu smo predstavili metode predvidevanja poplav. Opisali smo tudi tehnologijo LiDAR ter datotečni format, ki hrani zajete točke. Spoznali smo metode simulacij dinamike tekočin in njihovo uporabo v različnih aplikacijah ter na kratko predstavili že obstoječe implementacije metod SPH in SWE z različnimi pristopi. V poglavju 3. smo predstavili uporabljene tehnologije, kot sta Win32 in DirectX 10. Na kratko smo opisali postopek implementacije grafičnega pogona, branja vhodnih podatkov iz slikovnih datotek ali datotek LAS.

Med implementacijo algoritma smo se soočili z različnimi tehnikami predstavitve tekočine. Najprej smo poizkusili s preprosto valovno enačbo, ki pa se je izkazala za neuporabno za doseganje naših ciljev. Primorani smo bili poiskati alternative in smo tako spoznali odlično metodo simulacije tekočin imenovane enačbe plitve vode. Sprva je algoritem deloval le na enostavnih površjih, kar smo kasneje uspešno prenesli tudi na zahtevna površja, kot so na primer mesta. Algoritem smo preizkusili na različnih povšjih z različnimi vrednostmi vhodnih parametrov, kjer smo spremljali tok vode po terenu.

LITERATURA

[1] 1-D Saint Venant equation. Dostopno na: http://en.wikipedia.org/wiki/1-D_Saint_Venant_Equation. [18.08.2013].

[2] Weather forecasting. Dostopno na: http://en.wikipedia.org/wiki/Weather_forecasting. [23.05.2013].

[3] Windows API, Win32 API. Dostopno na: http://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx. [10.07.2013].

[4] DirectX Graphics and Gaming. Dostopno na: http://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx. [10.07.2013].

[5] Fluids 3.0. Dostopno na: http://www.rchoetzlein.com/fluids3/. [13.04.2013].

[6] GDI+. Dostopno na: http://msdn.microsoft.com/en-us/library/windows/desktop/ms533798(v=vs.85).aspx. [10.07.2013].

[7] Lecture 9: Lagrangian and Eulerian approaches. Dostopno na: http://nptel.iitm.ac.in/courses/103104043/Lecture_pdf/Lecture9.pdf. [10.08.2013].

[8] NVIDIA Cuda. Dostopno na: http://www.nvidia.com/object/cuda_home_new.html. [20.08.2013].

[9] Shallow water equations and the Ocean. Dostopno na: http://kestrel.nmt.edu/~raymond/classes/ph332/notes/shallowgov/shallowgov.pdf. [09.07.2013].

[10] Specifikacije datotečnega formata LAS. Dostopno na: http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html. [09.01.2013].

[11] Types of Flooding. Dostopno na: http://library.thinkquest.org/03oct/02054/floodtype.htm. [25.05.2013].

[12] Al-Sabhan, W., Mulligan, M., in A., Blackburn G. A real-time hydrological model for flood prediction using GIS and the WWW. *Computers, Environment and Urban Systems*, 27(1):9-32, 2003.

[13] Bridson, R. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, September 2008.

[14] Bridson, R. in Muller-Fischer, M. Fluid Simulation. Dostopno na: http://www.cs.ubc.ca/~rbridson/fluidsimulation/fluids_notes.pdf, Avgust 2007. [10.08.2013].

[15] Brodtkorb, A.R., Setra, M.L., in Altinakar, M. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers and Fluids*, 55:1-12, 2012.

[16] Crespo, A. J. C. *Application of the Smoothed Particle Hydrodynamics model SPHysics to free-surface hydrodynamics*. Doktorska dizertacija, ph. d. thesis,, University of Vigo, 2008. [05.06.2013].

[17] Geveler, M., Ribbrock, D., Göddeke, D., in Turek, S. *Lattice-Boltzmann simulation of the shallow-water equations with fluid-structure interaction on multi-and manycore processors*. Springer, 2011.

[18] Herault, A., Bilotta, G., Dairymple, R.A., Rustico, E., in Del Negro, C. GPUSHP: SHP free surface flow model for GPUs. Dostopno na: http://www.ce.jhu.edu/dalrymple/GPUSPH/Home.html. [02.07.2013].

[19] Kollinger, M., Vondraček, K., Šeblova, V., Zdražil, J., Jirka, J., in L., Vokounova. Flood simulation and visualization. In *Central European Seminar on Computer Graphics*, 2003. [13.07.2013].

[20] Lengyel, E. *Mathematics for 3D Game Programming and Computer Graphics*. Cengage Learning PTR, 3 izdaja, Junij 2011.

[21] Ma, Q. *Advances in Numerical Simulation of Nonlinear Water Waves*. World Scientific Publishing Company, April 2010.

[22] Petrie, G. in Toth, C. K. Introduction to laser ranging, profiling, and scanning. *Topographic Laser Ranging and Scanning Principles and Processing*, strani 1-27, 2009.

[23] Rustico, E., Bilotta, G., Herault, A., Del Negro, C., in G., Gallo. Smoothed Particle Hydrodynamics simulations on multi-GPU systems. Dostopno na: http://www.pdp2012.org/presentations/Feb16/Feb16_Rustico.pdf. [02.07.2013].

[24] Sir Lamb, H. *Hydrodynamics*. Dover Publications, 6 izdaja, Junij 1945.

[25] Thuerey, N. in Hess, P. Shallow Water Equations. Dostopno na: http://liris.cnrs.fr/alexandre.meyer/teaching/m2pro_charanim/papers_pdf/coursenotes_shallowWater.pdf. [06.05.2013].

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O AVTORSTVU

Spodaj podpisani/-a Matej Brumen

z vpisno številko E1041400

sem avtor/-ica diplomskega dela z naslovom:

Algoritem predvidevanja poplavnih območij z uporabo podatkov LiDAR

(naslov diplomskega dela)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  red. prof. dr. Borut Žalik

  in somentorstvom (naziv, ime in priimek)

  doc. dr. Domen Mongus

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela.
- soglašam z javno objavo elektronske oblike diplomskega dela v DKUM.

V Mariboru, dne 04.09.2013

Podpis avtorja/-ice:

Brumen

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O USTREZNOSTI DIPLOMSKEGA DELA

Spodaj podpisani/-a Borut Žalik (ime in priimek mentorja/-ice) izjavljam, da je

študent Matej Brumen (ime in priimek študenta/-ke) izdelal diplomsko

delo z naslovom: Algoritem predvidevanja poplavnih območij z uporabo podatkov LiDAR

(naslov diplomskega dela)

v skladu z odobreno temo diplomskega dela, Navodili za pisanje diplomskih del na dodiplomskih študijskih programih UM FERI in mojimi navodili.

Kraj in datum: Maribor, 04.09.2013

Podpis mentorja:

B. Žalik

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek avtorja-ice: Matej Brumen

Vpisna številka: E1041400

Študijski program: Računalništvo in informacijske tehnologije

Naslov zaključnega dela: Algoritem predvidevanja poplavnih območij z uporabo podatkov LiDAR

Mentor: red. prof. dr. Borut Žalik

Somentor: doc. dr. Domen Mongus

Podpisani-a MATEJ BRUMEN izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Zaključno delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija zaključnega dela je istovetna z elektronsko verzijo elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, varstva industrijske lastnine ali tajnosti podatkov naročnika: _____ ne sme biti javno dostopno do _____(datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela).

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum zaključka študija, naslov zaključnega dela), na spletnih straneh in v publikacijah UM.

Datum in kraj: 04.09.2013 Podpis avtorja-ice: Brumen

Podpis mentorja: _____
(samo v primeru, če delo ne sme biti javno dostopno)

Podpis odgovorne osebe naročnika in žig:
(samo v primeru, če delo ne sme biti javno dostopno) _____